Subject: Re: a serving of value_locate with a side of histogram
Posted by Jeremy Bailin on Mon, 22 Jun 2009 04:44:15 GMT
View Forum Message <> Reply to Message

On Jun 22, 12:06 am, Jeremy Bailin <astroco...@gmail.com> wrote:
> On Jun 21, 3:46 pm, Michael Galloy <mgal...@gmail.com> wrote:
>
>
>
>> David Fanning wrote:
>>> Jeremy Bailin writes:
>
>>>> Yeah, value_locate is very handy for problems like this! I
>>>> particularly like using it as a precursor to histogram - i.e. if you
>>>> want to do something fancy using reverse_indices but don't have
>>>> uniformly-spaced bins, first use value_locate to get integer indices
>>>> and then use histogram to do the heavy lifting.
>
>>> All right, I'll bite. Let's see an example of this.
>>> Maybe you can write an article and become the JD Smith
>>> of Value_Locate. :-)
>
>> No article, but I think this is what Jeremy is talking about:
>
>> IDL> ; get some random data
>> IDL> d = randomu(12345678L, 20)
>> IDL> print, d
>>   0.765989   0.0234537   0.589727   0.535102   0.982231
>> 0.693016   0.328147
>>   0.295642   0.849918   0.592262   0.558133   0.534926
>> 0.541119   0.594831
>>   0.410172   0.928598   0.161021   0.928724   0.952072
>> 0.522173
>
>> IDL> ; specify cutoffs
>> IDL> cutoffs = [0.3, 0.4, 0.8]
>
>> IDL> ; compute index of "bin" to put each value into
>> IDL> bins = value_locate(cutoffs, d) + 1L
>> IDL> print, ind
>>   2   0   2   2   3
>> 2   1   0
>>   3   2   2   2   2
>> 2   2   3
>>   0   3   3   2
>
>> IDL> ; compute histogram of bins
>> IDL> h = histogram(bins, reverse_indices=r)

```
>> IDL> print, h
>>           3           1          11           5
>
>> IDL> ; values less than 0.3
>> IDL> print, d[r[r[0]:r[1] - 1]]
>>      0.0234537     0.295642     0.161021
>
>> IDL> ; values between 0.3 and 0.4
>> IDL> print, d[r[r[1]:r[2] - 1]]
>>       0.328147
>
>> IDL> ; values between 0.4 and 0.8
>> IDL> print, d[r[r[2]:r[3] - 1]]
>>       0.765989     0.589727     0.535102     0.693016     0.592262
>> 0.558133     0.534926
>>       0.541119     0.594831     0.410172     0.522173
>
>> IDL> ; values greater than 0.8
>> IDL> print, d[r[r[3]:r[4] - 1]]
>>       0.982231     0.849918     0.928598     0.928724     0.952072
>
>> Mike
>> --www.michaelgalloy.com
>> Associate Research Scientist
>> Tech-X Corporation
>
> Yes, that's exactly the sort of thing. Here's another good example:
>
> Let's say you have 100,000 3D data points with each coordinate lying
> between 0 and 1, and you want to divide up the space into a grid
> 10,000 x 10,000 x 10,000 and determine which grid cells contain more
> than one point. It sounds like a good job for HIST_ND:
>
> h = hist_nd([[x],[y],[z]], min=0., max=1., nbins=10000)
> print, array_indices(h, where(h gt 1))
>
> The only problem is that it would require an array of one trillion
> elements that takes up 4TB of memory!
>
> But in reality, at least 99.99% of those cells must be empty. First,
> let's get a list of which cells contain any points:
>
> xbin = floor(x / 1e-4)
> ybin = floor(y / 1e-4)
> zbin = floor(z / 1e-4)
> bin = x + 10000ULL*(y + 10000ULL*z)
> sortedbin = bin[sort(bin)]
> uniquebins = sortedbin[uniq(sortedbin)]
```

>
> Now we can use uniquebins as a mapping function. The values of "bin"
> can range from 0 to 999999999999, but each one of those appears in
> "uniquebins", which has at most 100000 elements. So we replace each
> element in bin with *its location within uniquebins*:
>
> mappedbin = value_locate(uniquebins, bin)
>
> All elements of mappedbin are integers between 0 and 10000, which
> makes a perfectly reasonable histogram:
>
> h = histogram(mappedbin, min=0)
>
> The grid locations with multiple points are then:
>
> multipleindex = where(h gt 1, nmulti)
> if nmulti gt 0 then begin
>   xmulti = multipleindex mod 10000ull
>   ymulti = (multipleindex mod 100000000ull) / 10000ull
>   zmulti = multipleindex / 100000000ull
> endif
>
> ...and, of course, if you want to do anything with the data points
> that fall in those bins, you can do everything you'd normally do using
> reverse_indices.
>
> This is exactly the technique I used in WITHINSPHRAD_VEC3D from this
> discussion:http://groups.google.com/group/comp.lang.idl-pvwa ve/browse_thread/thr...
>
> I'll try to put together a bigger value_locate article later this
> week. :-)=
>
> -Jeremy.

I should learn to stop writing code at night. ;-) The grid locations
are really:

```
multipleindex = where(h gt 1, nmulti)
if nmulti gt 0 then begin
  xmulti = uniquebins[multipleindex] mod 10000ull
  ymulti = (uniquebins[multipleindex] mod 100000000ull) / 10000ull
  zmulti = uniquebins[multipleindex] / 100000000ull
endif
```

-Jeremy.